Dynamic Buffer Sizing and Pacing as Enablers of 5G Low-Latency Services

Mikel Irazabal, Elena Lopez-Aguilera, Ilker Demirkol, Senior Member, IEEE and Navid Nikaein

Abstract—3GPP standards organization is performing an impressive effort trying to reach sub-millisecond latencies for 5G. However, such efforts may become fruitless if exogenously generated delays at transport layer are not considered. Nowadays, Radio Access Networks (RANs) are deployed with large buffers to achieve full utilization and avoid squandering wireless resources. Unfortunately, and since the data path's bottleneck resides on the radio link, RAN's buffers are bloated by TCP's congestion control algorithm. Thus, a flow with low-latency requirements that encounters a bloated buffer, suffers from inevitable large sojourn times associated with the buffer depletion time, severely downgrading its Quality of Service (QoS). This paper presents different solutions for efficiently multiplexing distinct traffic patterns that share buffers on the 5G stack. Bufferbloat is extensively studied within the actual 5G QoS scenario, which presents multiple challenges inherited from the dynamic radio link nature and the presence of multiple queues at different entities. We propose and extensively emulate different algorithms in order to avoid the exogenous delay caused by the bufferbloat phenomena. We use real cellular network traces with realistic delay-sensitive and background traffic patterns in different scenarios. The outcome presents valuable insights in the algorithms that will enable low-latency services to be delivered through the 5G network stack satisfying restrictive envisioned constraints.

Index Terms—5G, SDAP, QFI, QoS, low-latency, BDP, Bufferbloat, BBR, AQM.

1 Introduction

Low-latency communications present a key use case in the 5th Generation (5G) cellular network standard. It is envisioned that many low latency services will successfully run on the 5G stack (e.g., tactile Internet, Vehicle-to-Everything (V2X) communications, online gaming, Voice over Internet Protocol (VoIP), etc.), meeting the different delay constraints these services require.

In order to achieve these goals, 3GPP is putting a remarkable effort trying to mitigate the endogenous causes that prevent sub-millisecond data delivery by engineering the channel access and designing new physical layer methods (e.g., mini-slots through new numerology, uplink grant free transmission instead of Scheduling Request procedure, pre-emptive scheduling for Ultra Reliable Low-Latency Communication (URLLC)). Additionally, Quality of Service Flow Indicator (QFI) has been introduced as 5G's finest grain QoS indicator [1], along with the new Service Data Adaptation Protocol (SDAP) sublayer [2] by 3GPP. However, no substantial effort has been invested in tackling exogenous causes (e.g., TCP's greedy congestion control algorithm), which are the governing factors of the end-to-end delay. In fact, 5G networks lack a solution to tackle the currently biggest challenge for ensuring a deterministic network latency: the bufferbloat [3]. The term was coined as the increased latency originated by the presence of excessive large (bloated) buffers in systems [4], and it is also a concern in other technologies that rely on QoS prioritization mechanisms (e.g., IEEE 802.1Q VLAN) [5]. In the presence of large buffers in the bottleneck data link, TCP's congestion control algorithm capability of estimating the available bandwidth gets distorted, and therefore, increases its sending rate until a packet is dropped, leading to a plethora of packets at the slowest link's buffer (i.e., the bottleneck) during the process. In order to overcome it, different approaches have been developed on the wired and IEEE 802.11 domains at various levels (e.g., Active Queue Management (AQM) algorithms, TCP Small Queues (TSQ) [6], TCP Segmentation Offload (TSO), Byte Queue Limit (BQL) [7], or new congestion control algorithms such as BBR [8]). However, not significant effort has been yet invested in carefully studying the 5G bufferbloat specifities. Thus, in this paper, we merge the most advanced principles applied nowadays for fighting the bufferbloat with the current state of the art in 5G low-latency solutions, and provide a thorough study of their effects on the 5G stack.

The bufferbloat in 5G networks is expected to specifically occur at the RAN since (i) RANs are nowadays deployed with large buffer capacities in order to compensate the data bandwidth variance caused by the physical radio channel; and, (ii) the packet forwarding speed capacities of the contemporary wired data transport technologies surpass that of the wireless technologies, effectively forming the data traffic bottleneck at the RAN. In fact, bulky services will try to monopolize the access to the wireless resources due to its inherited greedy TCP's congestion control nature, bloating the buffers on their way and impeding a rapid low-latency packet delivery of any other flow that traverses the same data path. On one hand, bloated buffers guarantee a full radio channel utilization. On the other hand, if in order to avoid the buffer bloating, the buffer capacity is restricted, a resource under-utilization scenario might

M. Irazabal (mikel.irazabal@upc.edu) and E. Lopez-Aguilera are with the Dept. of Network Engineering, Universitat Politècnica de Catalunya, Barcelona, Spain.
I. Demirkol is with the Dept. of Mining, Industrial and ICT Engineering, Universitat Politècnica de Catalunya, Barcelona, Spain.
N. Nikaein is with the Dept. of Communication Systems at Eurecom, Sophia-Antipolis, France.

happen due to the dynamic nature of the radio data link in cellular networks, which alters the radio channel capacity. Therefore, the objectives to fulfill are (i) to fully utilize the radio channel in order to avoid squandering wireless resources; and, (ii) to reduce the delay to the service time in order to avoid the queuing sojourn time.

In this paper, we devise 5G-compliant queue management solutions that achieve high throughput, while successfully delivering low-latency traffic[1]. This is achieved through the following contributions:

- We propose queue management solutions that avoid large queue sojourn times inherited from the bufferbloat problem considering (i) an extension of the SDAP sublayer; (ii) pacing capabilities; and (iii) TCP's transport layer congestion control algorithms (i.e., BBR and CUBIC).
- We extensively emulate and compare the proposed solutions using real cellular network traces with realistic delay-sensitive and background traffic patterns in distinct scenarios. The total radio link channel utilization and the low-latency flow delay are measured and analyzed.

The rest of the paper is structured as follows: Sections 2 and 3 describe the bufferbloat problem inherited from the actual 5G QoS scenario foreseen by 3GPP. The proposed solutions to tackle the bufferbloat are presented in Section 4. Section 5 describes the evaluation framework, while in Section 6 the emulated results are shown and analyzed. This paper ends with the conclusions in Section 7.

2 Background on 5G QoS model

The ambitious objective of 3GPP of a standard capable of successfully covering many use cases, inherently creates a complex QoS scenario that is meticulously described in [1]. In the following, we present the key aspects of such 5G QoS scenario for the downlink, while a similar approach applies to the uplink. Packets arrive from the Data Networks (DN) through the N6 interface to the User Plane Function (UPF), where the first buffer that a data packet will encounter in its path to the User Equipment (UE) is located. The UPF identifies and segregates these data flows based on the configuration received from the Session Management Function (SMF). SMF is responsible for, among other functions, the UE IP address management, the control part of policy enforcement and QoS, and managing the Packet Forwarding Control Protocol (PFCP) session. PFCP session describes how packets should be identified and marked with its QFI through the Packet Detection Rule (PDR), policed through the Multi-Access Rule (MAR), forwarded based on the Fowarding Action Rules (FARs), tagged based on the QoS Enforcement Rules (QERs) and lastly reported using the Usage Reporting Rules (URRs) [1] [10]. This is

1. We study the bufferbloat phenomenon as the main 5G exogenous delay cause. The equally important 5G endogenous delays (e.g., non-scheduled access [9]), are out of the scope of this paper.

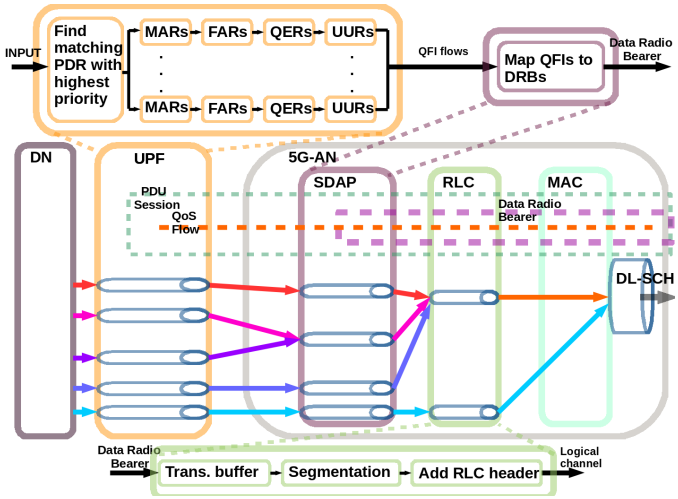

Fig. 1. Simplified 5G QoS downlink block diagram with the entities with buffers and PDU session, QoS flow and Data Radio Bearer abstractions.

the first entity in the 5G downlink scenario where traffic engineering techniques and packet buffering take place as depicted in Fig. 1.

QFI is a 6 bit field (i.e., $2^6 = 64$ indicators with different priorities can be defined). Every QFI is associated with a different characteristic according to [1] among which, the maximum data burst, the resource type, the priority level, the Packet Delay Budget (PDB) or the Packet Error Rate (PER) can be highlighted. Three different resource types are described: Guaranteed Bit Rate (GBR), Non-Guaranteed Bit Rate (Non-GBR) and Delay-Critical GBR [1]. The priority level indicates the weight of the packet for scheduling purposes. The PDB is the upper bound delay permitted, measured from the N6 interface until the UE [1]. It is also taken into account to determine the scheduling weight and HARQ retransmissions [1]. Although such measurement covers the 5G network, it does not consider the end-to-end delay of the user application. The PER is defined as the number of packets processed by the RLC sublayer of the sender, divided by the number of packets delivered to the PDCP sublayer at the receiver. According to [1], a single UPF or multiple UPFs can be provided for a given PDU session. The UPF selection is decided by the SMF. If deterministic low-latency is required, the UPF will be created as close as possible to the 5G Access Network (5G-AN). Software Defined Networks (SDN) will provide the possibility of instantiating and scaling entities "on the fly", therefore enabling the relocation of UPFs, and thus reducing the latency, as demonstrated in [11] and described by ETSI in [12].

Packets will then arrive to the SDAP sublayer at 5G-AN [2] (c.f. Fig. 1), where the second buffer that a data packet will encounter in its path to the UE is placed. A SDAP per PDU Session is described in [2], even though it is mentioned that other implementations are possible. RRC configures this sublayer to map the QFIs assigned by UPF into Data Radio Bearers (DRBs)

[13]. A maximum of 64 QFIs and 8 DRBs are allowed per UE [13]. Therefore, a many to one mapping will inevitably occur at the SDAP sublayer and data packets marked with different QFIs will inevitably share queues. 3GPP defines the SDAP *raison d'être* as a simple mapper. This implies that a First In First Out (FIFO) structure is foreseen since no scheduling requirement is enabled. However, if heterogeneous traffic delay and priority constraints must be met, mobile network operators will deliver a packet scheduling solution that can selectively forward packets with different requirements, since once a packet is assigned to a queue, segregating it according to their QFI can be costly and complex. Moreover, if a scheduler is not implemented at the SDAP sublayer, the only possibilities for traffic engineering where buffering occurs is the UPF [10] and the RLC [14] as seen in Fig. 1. Since the UPF network function and the RLC sublayer are not contiguous, the delay for communicating both entities may be fatal if 5G low-latency requirements must be met. Therefore, we enhance the standard SDAP with different queues according to the tagged QFI, and add scheduling capabilities in order to achieve a more fine-grained packet forwarding mechanism.

Once the packets are mapped into a DRB, they are forwarded to the RLC sublayer, where data is buffered, segmented and the RLC header is added in the transmission side. It has to be noted that there will be a RLC entity per UE and DRB. This is the last queue foreseen by 3GPP since MAC is not provided with one, and we can neglect the HARQ queue since it does not consume data but rather retransmits it.

3 Related Work

In 5G, despite of the low-latency requirements [1], unfortunately no substantial effort has been invested in understanding and reducing the bufferbloat problem. This will be of outstanding importance for next generation services that require low-latency in order to function correctly (e.g., tactile internet, VoIP, online gaming, etc). New initiatives (i.e., the Open RAN (O-RAN) [15] alliance) are enabling the possibility of embedding intelligence into the RAN, aiming to improve the latency requirements of different services. Data traffic patterns are predicted and actions are taken accordingly (e.g., through dual connectivity). However, such approach does not scale once the traffic exceeds certain threshold. Additionally, slicing has emerged as a possible solution to serve low-latency traffic, where the flows are segregated according to their requirements [16]. Nonetheless, due to the finite number of DRBs (i.e., 8 per UE), flows will start sharing buffers along the data path once their number exceeds the maximum amount of available DRBs, following the pigeon hole principle. Moreover, the flow to DRB assignment is not a trivial task. Hence, the default assignment of non-classified flows is to the Best Effort DRB. Additionally, in order to maximise revenue, operators will need to maintain the queues uncongested to accept as many services as possible, fulfilling their latency requirements.

3.1 Tackling the bufferbloat phenomena from the queues perspective

In order to fight against the bufferbloat problem in the Internet routers, AQM was developed. A natural deployment point for AQM in 5G is the RLC sublayer, as it is the last buffer before the data link bottleneck (i.e., the wireless data link is considerably slower than the wired data link). In [17], applying the RED algorithm [18] at RLC is proposed. RED discards packets probabilistically according to the actual average number of packets in the buffer. When the buffer is empty, RED accepts all the incoming packets. As the buffer occupancy increases, so does the probability of rejecting the following incoming packets. If the buffer reaches the full state, the following packets are rejected. Since most of the TCP's congestion control algorithms' implementation deliver their packets in bursts, RED believes that the queue is approaching to a congestion state, its packet discard probability increases and, consequently, it starts discarding packets misinterpreting the congestion state. In addition, RED needs some tuning parameters, which made it effectively useless if the traffic patterns were not carefully studied beforehand, avoiding its wider adoption.

CoDel [19] is a newer AQM algorithm that aims to improve RED's shortcomings. CoDel is a packet sojourn time based algorithm that discards packets in order to inform the sender's TCP congestion control algorithm that excessive buffering is taking place. It is one of the best known AQM, and a widely implemented solution to tackle the bufferbloat problem [20]. It is governed by two variables, the *interval time* and the *target time*. CoDel inserts a timestamp in every packet that is enqueued. During the *interval time*, the sojourn time of every egressed packet is measured, and the minimum value is saved. If after the *interval time*, the minimum saved value is above the *target time*, the next packet is dropped as a mechanism to inform the sender TCP flow that excessive queuing is happening. The *interval time* is then decreased by $1/sqrt(x)$, where x starts at 2 and increases every interval where the desired sojourn time is not reached. The *interval time* is recommended to be configured at 100 ms, while the *target time* is recommended to be a 5% of the *interval time*. More details can be found in [19]. In conjunction with a modified Round Robin scheduler known as Deficit Round Robin [21], the FlowQueue-CoDel Packet Scheduler (FQ-CoDel) [22] is formed. FQ-CoDel is nowadays the AQM algorithm activated by default at some embedded router projects [20]. In the 5G context, FQ-CoDel has been mentioned in [23]. There, a FQ-CoDel entity is proposed at the UPF network function. In order to generate the bottleneck at the UPF, so that FQ-CoDel can effectively work, the Round-Trip Time (RTT) of the packets is measured, and the optimal egress rate per flow is calculated. Packets are then delivered at a slightly smaller transmission

rate than the calculated rate, effectively moving the bottleneck from the 5G-AN to the UPF. Unfortunately, no results are provided for such an approach. Moreover, this method presents some weak points. Firstly, 5G data rate changes abruptly due to the radio channel conditions. If the amount of resources for one UE fluctuates, the UPF will notice such change with a considerable delay, during which the channel will be underutilized or queues will start forming. Additionally, the information will be influenced by the downlink and uplink delay, which are asymmetric and non-negligible. Moreover, some newer protocols (e.g., QUIC, SCTP) that do not rely on feedback from the transport layer protocol, but from the application level, cannot be correctly managed with such an approach. Hence, such an scenario has been discarded in our work.

Traffic with different constraints needs to be segregated and scheduled accordingly in order to meet their requirements correctly. Higher priority traffic should avoid the large sojourn times that are endogenous to the phenomenon of sharing queues with bursty flows. Fairness also plays a major role since flows transporting big data quantity should not monopolize the access to transmission resources. In order to address this problem, one of the first algorithms developed was the Stochastic Fair Queuing (SFQ) [24]. The SFQ segregates the arriving flows with a hashing function and forwards them to a statically allocated array of queues. A Round Robin scheduler egresses later the packets fairly among each active queue. Since hashing collisions may occur, SFQ periodically alters a perturbing value that is used in the hashing process in order to avoid a possible hash collision. SFQ at the PDCP sublayer together with a self developed algorithm has been explored in [25] with promising results. There [25], a dynamic RLC queue is implemented according to the delay that packets experience at the RLC queue. If the delay is above the target delay, the queue capacity is shrunk; however, if the delay is below the target delay, the queue capacity is increased. In order to operate correctly, a cross communication between the PDCP and the RLC sublayers is needed. The algorithm is presented under the name of DynRLC and it is successfully tested on the open source LTE OpenAirInterface [26] project.

Another important mechanism to manage the low-latency requirements explored in the literature is the limitation of the queues. Since packets that are aggregated into one queue are treated equally, maintaining the queues as empty as possible enables the possibility to avoid big latencies associated with big queue sojourn times. In recent years, this principle has been consistently applied at different levels in the Linux kernel's TCP/IP stack with great success [6] [7], as well as in the radio framework as presented in [27] [28], and in our preliminary work [29]. In [27], the traffic control mechanisms provided by the Linux kernel stack applied to the cellular context are analyzed. Such an approach exploits the fact that the worldwide popular Android OS is based on the Linux kernel in order to propose a realistic solution. Network traffic will flow through the Linux kernel's IP stack where the traffic control (tc) mechanisms can be applied, among which the BQL [7] algorithm is tested. The BQL algorithm resizes the queue to its optimal byte size according to the last egress rate. Such a dynamical mechanism has proved to be adequate since a static small queue may reject packets before achieving the full transmission rate capability [30]. Even though good simulation results were obtained, the study considers the cellular access network as a queue, while in reality, the QoS queuing is composed of multiple hierarchical queues [1] (c.f. Fig. 1). All the traffic inside the access network is treated equally, which deprives the possibility of a finer grained segregation and more refined QoS guarantee. In [28] a *Q-learning* algorithm for limiting the MAC queues of the IEEE 802.11 protocol is presented under the name of *LearnQueue*. While very interesting results are obtained through a backpressure mechanism, such design is not possible at 5G since its MAC sublayer is devoid of any queue. Additionally, a reinforcement learning method, even though it can theoretically maximize the total reward needs significant exploration time. If a new actor with not foreseen characteristics joins in, the algorithm needs some time until it can correctly assign the resources. In an heterogeneous scenario as 5G, this can lead to a fatal transition time until the service can correctly be served. In any case, such an approach definitely deserves more research on the 5G field. The benefits of AQM algorithms inside the 5G stack are also analyzed in our preliminary work [29]. However, even though the problem is well identified, it lacks from a dynamic solution that can fit in nowadays cellular networks. Therefore, in this paper we propose, implement and analyze different novel solutions, as well as the ones proposed in the literature [7] [19] [25], and tackle the inherent dinamicity problem of 5G networks with different queueing levels and real traffic patterns.

3.2 Tackling the bufferbloat phenomena from the congestion control algorithm perspective

Some UE vendors rely on limiting the congestion window in order to inform TCP's congestion control algorithm that congestion is happening, and therefore, to avoid excessive bufferbloat in their network [31]. A more refined way is the new TCP BBR algorithm [8]. BBR is based on the Bandwidth Delay Product (BDP), as opposed to loss-based congestion control algorithms. The data in-flight is calculated as the result of multiplying the Round Time Propagation (RTprop) and the Bottleneck Bandwidth (BtlBw) available for each flow. The BtlBw assures that the full link is utilized, while the RTprop guarantees enough in-flight data to prevent queue starvation on the bottleneck link. BBR [8] firstly measures the available bandwidth through the increase of RTT while pumping packets. After saturating the data link, it starts a depletion process in order to decongest

the buffers that were congested during the first BDP estimation. Once in steady state, it periodically sends more packets to verify that it is working at the maximum BDP. If the RTT increases, it can be translated as the result of the self created congestion, and a depletion process is followed. BBR updates its RTprop and its BtlBw every received ACK. On one hand, RTT is composed of RTprop + η. RTprop is a physical network property that just changes with the path, while η is mostly affected by the sojourn time experienced at the queues. Therefore, the lowest RTT is the value that more closely resembles to RTprop. This value is taken as the RTprop used for calculation over a window time which can typically vary from tens of seconds to minutes. On the other hand, the delivery rate can be calculated as delivery_rate = Δdata_delivered/Δt. In any case, the delivery_rate can never surpass the BtlBw, therefore, BBR updates BtlBw to the maximum value observed during a window time which is typically six to ten RTTs. In this way the values of the RTprop and BtlBw are dynamically adapted to possible data link alterations. In order to assure that such values are correct, every 10 seconds a 200 ms timeframe[2] is utilized for recalculation. Lastly, BBR transmits the packets through a pacer instead of in bursts in comparison to other TCP congestion control algorithms (e.g., CUBIC), since it tries to match its egress rate to the actual bottleneck bandwidth [8] without creating queues.

4 Proposed low-latency queue management algorithms

As previously mentioned, any proposed solution should fully utilize the radio link channel, while reducing the delay to the service time. Due to the dynamicity in 5G's data link bandwidth and the data traffic patterns, the following challenges stem from the aforementioned objectives:

- Estimating the time-variant optimal buffer occupancy for SDAP and RLC sublayers.
- Delivering the packets from the UPF/SDAP entities to the SDAP/RLC sublayers at the optimal pace.

5G cellular networks consist of at least three levels of queuing: RLC sublayer, SDAP sublayer and UPF network function buffers. Every buffer must always contain enough bytes to serve the next entity request, but any additional byte will just cause delay and would better be kept at upper layers. SDAP and RLC buffers need to dynamically adjust its capacity according to the data link bandwidth (i.e., the radio link capacity and the number of Resource Blocks (RBs) assigned). Additionally, in an optimal scenario, packets should flow through a pacing algorithm as such a behaviour maximises the possibilities of avoiding a congested buffer. In the next subsections we propose and analyze different solutions that alleviate the actual bufferbloat phenomena at RLC and SDAP sublayers, enabling low-latency time constraint services to successfully share the 5G stack with other services.

2. These values are taken from the Linux kernel version 4.15.0-60-generic.

4.1 Proposed RLC queue management algorithms

As the wireless domain is significantly slower than the wired domain, the bottleneck, and therefore the bufferbloat, in actual cellular network systems resides at the entity that holds the last buffer in the wired domain (i.e., RLC buffer). Thus, a solution for tackling the bufferbloat problem in 5G must inevitably start reducing the buffers' occupancy at RLC. A naive solution at RLC sublayer would assign a slightly superior buffer size per DRB than the maximum possible delivery rate to every UE under the best radio conditions, and if it is surpassed, packets will be dropped. Such a solution is implemented in OpenAirInterface [26]. However, such an approach creates big sojourn times at the RLC buffers if the MAC scheduler assigns a partial amount of available RB to the UE (since multiple UEs compete for the available bandwidth), or if suboptimal radio channel conditions occur. Additionally, the foreseen 5G softwarization capability will be other factor to consider as it may also alter the resources assigned to a UE abruptly (e.g., through slicing). Such a problem is also present at the Linux kernel IP stack. Before the Network Interface Controller (NIC), there is the so-called driver queue, where packets are accumulated and from which the NIC is fed. On one hand, if the NIC wakes up and tries to pull data off the driver queue but there are no packets in it, a transmission opportunity will be squandered and the throughput will decrease. On the other hand, if too many packets are present in the driver queue, a large sojourn time will occur. In this scenario, Dynamic Queue Limit (DQL) [32], which is an implementation of BQL, appeared with the aim of limiting the number of bytes in the driver queue without starving it and avoiding excessive packet accumulation. After an interval of time, DQL assesses whether the hardware was starved and the queue limit was reached, in which case, the queue limit is increased. If there are still bytes to transmit in the queue, the queue limit is decreased by the number of bytes not transmitted yet [32]. Taking this into account, and since the RLC buffer, together with the driver queue, is the last buffer in the data path, we propose the *Dynamic RLC Queue Limit (DRQL)*.

The DRQL algorithm (c.f. Algorithm 1) consists of the following variables: *limit* that represents the queue limit in bytes, *dequeued_bytes* that provides the bytes that were forwarded, *last_time* that is a timestamp since last interval, T that represents an interval during which the sojourn time of the packets is measured, and *min_val* that saves the minimum sojourn time during an interval. During initialization, T is set to the system Transmission Time Interval (TTI), *min_val* to the maximum value supported by the type of the system, *dequeued_bytes* to 0, *limit* to the maximum RLC queue capacity and *last_time* to the actual time. The algorithm works in

Algorithm 1 DRQL: Dynamic RLC Queue Limit pseudo-code

```
1: T ← TTI, min_val ← INF, dequeued_bytes ← 0
2: limit ← MAX_VAL_LIMIT, last_time ← now
3: procedure DEQUEUED(bytes)
4:     dequeued_bytes ← dequeued_bytes + bytes
5:     update limit_reached
6:     update remaining
7:     buffer_starved ← no remaining AND limit_reached
8:     if buffer_starved then
9:         increase buffer's limit
10:    else if remaining then
11:        if min_val greater than remaining then
12:            min_val ← remaining
13:        if last_time + T greater than now then
14:            last_time ← now
15:            limit ← limit − min_val
16:            min_val ← INF
17:    else if last_time + T greater than now then
18:        last_time ← now
19:        min_val ← INF
```

the following way: on one hand, the SDAP sublayer is responsible for continuously querying the $limit$ value to the RLC, as well as the actual size, and deciding whether to send more bytes or keep them. On the other hand, the MAC sublayer is responsible for calling the *DEQUEUED* procedure whenever it dequeues data from the RLC buffer, which will happen under hard real-time constraints as MAC will call the procedure every TTI. The *DEQUEUED* procedure first checks if the limit was reached during that interval of time (line 5), and whether there are remaining packets at the queue (line 6). Next, it checks whether a queue starvation happened (line 8). If this is the case, the buffer limit is increased as starvation happened and some transmission possibilities were squandered. If the buffer limit was reached, and no more data remains at the queue (definition of *buffer_starved*, line 7), the queue could had been provided with more data, and some bandwidth has been squandered. Therefore, the buffer limit is immediately increased. If, on the contrary, some data still remains in the buffer (line 10), and it is smaller than the *min_val*, this value is assigned to *min_val*. If after an interval time (line 13), there has always been remaining data, the *limit* value is reduced by the lowest value observed during that interval, and kept at *min_val*. If the limit is not reached and there is no remaining data (i.e., when the number of bytes that arrived at the RLC buffer were not enough to start accumulating), the internal timer is simply reset and the *min_val* is set to *INF* as all the data that was forwarded into the buffer on that interval was successfully delivered to the PHY layer.

RLC's optimal queue size can also be modeled using queuing theory as depicted in Fig. 2, where the SDAP and the RLC sublayer queues are involved. Based on queuing theory from [33] and [34], and using 5G's specificities, we can model λ as the arrival rate, μ as the service rate of each server and K as the number of servers in the system. Additionally we define $\rho = \lambda/(K\mu)$ as the utilization factor. The Little's Theorem [34] (for deterministic as well as stochastic flows) determines that the average number of customers in the queue (i.e., $\overline{N}$) results from the equation

$$\overline{N} = \lambda T(\rho) \tag{4.1}$$

where $T(\rho)$ is the mean system response time (the sum of the sojourn time at the queue and server time). Since server time cannot be reduced, the minimum mean response time occurs when the sojourn time is 0, which happens when there are no customers in the queue and, therefore, $\rho = 0$, which implies that $T(0) = 1/\mu$. If we consider a deterministic arrival and a deterministic service, we can model our system as a D/D/K system. This is a simplification in a 5G system, but the deterministic approach is very useful due to the fact that according to the Law of Large Numbers [34], some conclusions may be extended to stochastic systems. As there are K servers, the total system service capacity is equal to $K\mu$ and, therefore, $\lambda_{\text{max}} = K\mu$ if the sojourn time is to be minimized. The behaviour of the system can be graphically understood looking at Fig. 3, where there exists two inaccessible regions. On one hand, no matter how many customers get in the system, the utilization factor ρ cannot surpass full normalized utilization (i.e., the region on the right of $\rho(1.0)$). If the arriving customers rate (λ) is superior to the service time (μ) times K (i.e., the serving rate), the mean response time ($T(\rho)$), tends to infinity as customers start accumulating at the queue. On the other hand, the mean response time of the system ($T(\rho)$) cannot be reduced beyond the serving time (i.e., the region below $T(0)$). With this constraints, the optimal point is the β knee (with $\rho = 1$, $T(\rho) = T(0) = T(1.0)$), where the response time is the minimum while the utilization is maximum. From (4.1), for the optimal operating point with $\lambda_{\text{max}} = K\mu$, we get the intuitive result of $\overline{N^*} = K$ (i.e., no more customers than servers). Conversely, the BDP (the BDP is in fact the optimal theoretical point where no queue is generated while enjoying full bandwidth as demonstrated by [35]) is defined as the *Bottleneck Bandwidth (BBandwidth)* times the *No-Load Delay (NLDelay)* (i.e., the time needed to physically traverse the path). In this case, the *NLDelay* is $1/\mu$ and the *BBandwidth* is $K\mu$, and therefore, the $BDP = K$, which results in $BDP = \overline{N^*}$. This is a very valuable result that can be intuitively well understood. We want to maintain our server fully utilized, while the customers should suffer zero queue sojourn time. We are applying the principle of *keeping the pipe full, but not fuller* meticulously described by Kleinrock at [33].

In 5G systems, on one hand the radio channel quality is delivered through the Channel Quality Indicator (CQI) in uplink to the 5G-AN by the UE. The CQI index is a scalar, the value of which is translated into a modulation (i.e., QPSK, 16QAM, 64QAM or 256QAM) and from it, to a Modulation and Coding Scheme (MCS) index [36]. The MCS index is then converted to the total number of transport bytes, depending on the RBs assigned by

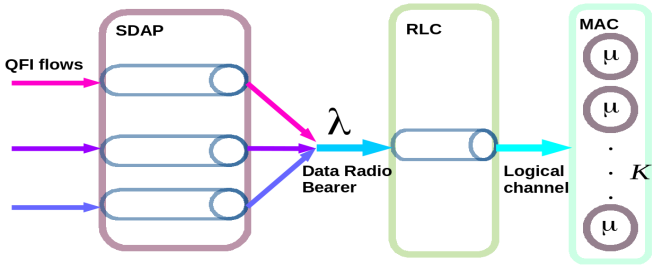

Fig. 2. Simplified data path model for SDAP and RLC sublayers queues with multiple QFIs that converge to one Data Radio Bearer.

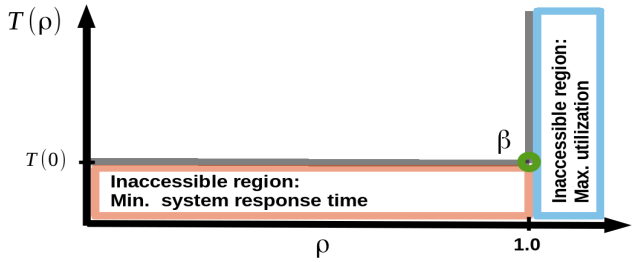

Fig. 3. A D/D/K Deterministic Queuing System.

the MAC, to assure an error rate lower than 10% [36]. With this metric, the total radio data link bandwidth can be computed. On the other hand, the radio slot length is known, and since the serving time of the packets is clearly governed by the slot duration (the physical propagation delay and the processing time of the packets can be neglected) the delay metric can be obtained. In 5G, the slot duration can vary from 1 ms to 62.5 μs at the cost of utilizing more frequency spectrum (i.e., from 15 kHz to 240 kHz) [37]. Since the lowest common denominator is 1 ms for all the slot durations, we can calculate the optimal BDP every 1 ms by multiplying the maximum slot duration (i.e., 1 ms) with the data link bandwidth. In essence, every 1 ms a new D/D/K queuing system is calculated, where every server μ, can process the minimum number of information (i.e., one bit) per millisecond, and the number of servers K varies with the time according to the aforementioned conditions, with a serving time equal to the maximum radio slot duration (i.e., 1 ms). Modeling the arrival and the service time as deterministic is a simplification. The various retransmission mechanisms (i.e., HARQ/NACK and RLC AM) or an abrupt change in the number of RBs available per RLC buffer can impact the sending rate. However, the conclusions obtained from deterministic model assumptions have already shown their benefits in real network (i.e., stochastic) deployments. For example, BBR [8] uses a deterministic queue to model the network's behaviour even though the network bandwidth and RTT are non-deterministic (e.g., a path may be shared with other flows, altering the available bandwidth and RTT). The continuously newly estimated BDP, eventually determines its packets forwarding pace.

Algorithm 2 5G-BDP: 5G Bandwidth Delay Product pseudo-code

```
1:  offset ← NORMALIZED_OFFSET
2:  bytes_tx ← 0, bytes_to_send ← 0, last_time ← now
3:  procedure SET_VALUES (remaining, channel_capacity)
4:      if channel_capacity greater than remaining then
5:          bytes_to_send ← channel_capacity − remaining
6:      else
7:          reset bytes_to_send
8:      reset bytes_tx
9:      last_time ← now
10: function GET_OPTIMAL_VALUE
11:     update elapsed_time
12:     update normal_tti
13:     normal_tti ← normal_tti + offset
14:     update soll_tx
15:     if soll_tx greater than bytes_tx then
16:         return soll_tx - bytes_tx
17:     else
18:         return 0
```

Aiming to work on the optimal BDP operation point, we present a cellular BDP algorithm denoted as *5G-BDP* (c.f. Algorithm 2). During the initialization, the variable *offset* is set to a value in the range of [0,1.0). This value is aggregated to the normalized TTI value in order to reduce the starvation possibility (line 13). The normalized TTI represents the normalized time elapsed since last TTI. The variables *bytes_to_send* and *bytes_tx* are set to 0 and the *last_time* to *now*.

MAC sublayer calls the *SET_VALUES* procedure every TTI to calculate the optimal number of bytes to forward (i.e., *bytes_to_send*), to reset the number of packets transmitted in the last interval (i.e., *bytes_tx*) and to save the actual time. For this, the CQI index received from the UE together with the last number of used RBs are converted into bytes, which is passed to the function through the *channel_capacity* variable together with the remaining number of bytes at the RLC buffer (i.e., *remaining*). The number of remaining bytes in the queue informs the algorithm how many bytes are accumulated at the RLC buffer in order to predict the optimal quantity of bytes to forward.

In order to forward the packets at a correct pace from the SDAP buffer to the RLC buffer, the SDAP sublayer calls the *GET_OPTIMAL_VALUE* function periodically with a period smaller than a TTI. It calculates the normalized value of the time elapsed since the last time that new values arrived (i.e., *elapsed_time*) at the *SET_VALUES* procedure, and determines how many bytes could had been forwarded from the SDAP towards the RLC within this duration (i.e., *soll_tx*). If the amount of already transmitted bytes exceeds this latter value, no packets are forwarded to the RLC buffer. Otherwise, the difference between the number of bytes that could have been transmitted and the actual amount of transmitted bytes is returned (line 16). With this information, the SDAP determines how many packets can be dequeued. There might exist a difference between the estimated BDP and the real BDP due to divers factors as previously men-

tioned. 5G-BDP minimizes the impact of this difference, by using the remaining number of bytes at the RLC buffer to calculate the $bytes\_to\_send$ every TTI. In this way, the queue formed when the BDP is overestimated (i.e., the real BDP is smaller than the calculated BDP and more data than the data pulled by the MAC has been forwarded), is reduced in the next TTIs, as no more data will be forwarded until the buffer is depleted below the $channel\_capacity$ as described at Algorithm 2 in procedure SET_VALUES.

5G-BDP differs in various ways from DRQL. DRQL defines a limit, which controls the amount of data to pass to the RLC. However, unlike 5G-BDP, this limit is not updated based on the BDP. Instead, it uses a heuristic to update that limit value based on the buffer state. 5G-BDP on the contrary uses the channel capacity value and the RLC buffer state to control the SDAP-RLC data transfers. In addition, 5G-BDP uniformly distributes the sending of packets to the RLC within a TTI, which defines a pacing mechanism at SDAP allowing prioritization of low-latency traffic arriving during that TTI. To do this, 5G-BDP forwards packets according to the number of bytes that could had been used for the elapsed time since last TTI (e.g., if 0.5 ms elapsed since last TTI and the BDP is 2200 bytes/ms, 1100 bytes can be forwarded).

4.2 Proposed SDAP queue management algorithm

If a containment mechanism is implemented at the RLC buffer, the bufferbloat will be transferred to the next queuing level as demonstrated at [29]. If, on the contrary no containment mechanism is implemented at the RLC sublayer, any attend to tackle the problem at the SDAP will be superfluous. Therefore, any queue management algorithm proposed at the SDAP sublayer must be combined with a containment mechanism at the RLC in order to be effective.

For scenarios where a quick communication in comparison with the TTI between the UPF and the 5G-AN can be established, we present the *UPF-SDAP Pacer (USP)* algorithm. USP forwards the packets from the UPF to the SDAP at a rate that avoids building large queues at the SDAP sublayer, while maintaining its buffer always occupied. Algorithm 3 illustrates our proposal. T represents a time interval, and num_ticks is the number of times that the UPF queried the SDAP sublayer during the last T interval. The number of dequeued bytes so far is saved at $dequeued\_bytes$, while the amount of dequeued bytes at the last T interval are saved in the $dequeued\_bytes\_last$ variable. The $saturation\_detected$ represents a flag to inform if the RLC buffer stopped accepting data, and the $optimal\_occ$ represents the number of optimal number of bytes that the SDAP buffer should have in every moment. The UPF network function asks to the USP regarding SDAP's actual size, as well as the optimal SDAP occupancy (line 17). In case that the actual occupancy is below the optimal occupancy, enough packets to reach the optimal occupancy

Algorithm 3 USP: UPF-SDAP Pacer

```
1: T ← TTI, num_ticks ← 0, dequeued_bytes ← 0
2: dequeued_bytes_last ← 0
3: saturation_detected ← False
4: optimal_occ ← MAX_OCC
5: procedure DEQUEUED(bytes)
6:     if saturation_detected then
7:         saturation_detected ← False
8:         calculate optimal_occ
9:         last_time ← now
10:        dequeued_bytes_last ← dequeued_bytes
11:        reset num_ticks
12:    else if now greater than last_time + T then
13:        last_time ← now
14:        dequeued_bytes_last ← dequeued_bytes
15:        reset num_ticks
16:    dequeued_bytes ← dequeued_bytes + bytes
17: function GET_OPT_OCCUPANCY
18:    increment num_ticks
19:    if now greater than last_time + T/2 then
20:        if once per T then
21:            optimal_occ ← clamp(2 * optimal_occ)
22:    return optimal_occ
```

are sent. Every time that the UPF executes a query to get the optimal occupancy (line 17), the *num_ticks* value at USP is increased (line 18). It has to be noted that this value may suffer deviations in soft real-time environments, since the deadlines are non-deterministic in such environments. On the other hand, the SDAP sublayer will set the *saturation_detected* flag to $True$ once the RLC buffer reaches a congested state. Once the RLC buffer accepts more data again, SDAP will dequeue data from its queue and the *DEQUEUED* procedure (line 5) of the USP algorithm will be called. The *DEQUEUED* procedure will compare whether a saturation state was reached in the previous interval (line 6). If this is the case, the flag will be reset and a new optimal occupancy value for the SDAP queue will be calculated (line 8) according to the bytes accepted by the RLC buffer in the previous interval and the number of times that UPF asked to the SDAP for the *opt_occupancy* of the queue (lines 17 - 22). The optimal occupancy of the queue is doubled after half of the TTI is consumed (i.e., 0.5 ms for 4G systems where the TTI last 1 ms) (line 19), adopting a more aggressive pacing strategy than 5G-BDP. Note that it will just be doubled once every T. This mechanism is an heuristic that assures feeding the SDAP sublayer with enough bytes, even in the case where, due to the soft real-time environment, the query time from the UPF may suffer deviations. It also assures that in case that no saturation was detected in the *DEQUEUED* procedure, a saturation state will be found in the near future if enough packets are forwarded. The *optimal_occ* value is clamped not to grow indefinitely in case where no saturation is reached or to fall to zero in case that no packets were dequeued.

This algorithms differs from the 5G-BDP in various ways, even though the idea of maintaining the queue size *full but not fuller* [33] remains. In the first place, it is of vital importance not to starve the SDAP queues,

since that may also starve the RLC queues and ultimately reduce the bandwidth. Therefore, in a soft real-time communication system (i.e., UPF may not always have the same amount of possibilities to transmit packets to the SDAP), as the one between the UPF and the SDAP, USP carries a more aggressive packet forwarding pacer than 5G-BDP, with the objective of maintaining the SDAP queue at an optimal value, but never to eradicate it completely. Secondly, no beforehand information about the bandwidth is shared (e.g., CQI index). USP is not provided with more information rather than the actual status of the SDAP queues, and the number of times that UPF queried the SDAP sublayer during the last interval time. With this data it calculates the optimal queue occupancy at the SDAP sublayer.

5 Evaluation Framework

In order to validate and evaluate the proposed algorithms, we developed a hierarchical queuing system[3] that emulates the queueing scenario described by 3GPP [1]. A comparison of the 5 algorithms (i.e., 5G-BDP, DRQL, DynRLC, CoDel and USP) used in this work is provided in Table 1, where the main characteristics of the algorithms can be observed. In order to optimize the number of packets at the RLC queue, the cross-layer communication plays a major role as the information can rapidly flow between close entities. This characteristic makes 5G-BDP, DRQL and DynRLC significantly faster in comparison with BBR. As previously mentioned, a pacing algorithm augments the possibilities of delivering a high priority packet within the actual TTI, giving an advantage to 5G-BDP over their competitors. CoDel is announced as a knob-less solution. In the CoDel RFC [19], it is recommended to set the *interval time* value to 5 ms, although no theoretical background for such number is given but rather a heuristic. Even though such value has been proved to work correctly on the wired domain and in some open source IEEE 802.11 implementations has been adopted as the default AQM algorithm [20], it does not address 5G specificities (e.g., the *interval time* or the *sojourn time*). In the same way, there are some variables that need to be preconfigured at DynRLC for a correct functioning. Assigning correctly its values in a highly dynamic scenario such as 5G, is a non trivial task that still remains open. We also present USP, which is an algorithm to be placed between the UPF and the SDAP entities. USP was designed with the same principle as 5G-BDP (*keep the pipe full, but not fuller*), and, therefore, it shares its main characteristics with 5G-BDP. Lastly, CoDel is the only algorithm that discards packets to inform the congestion control algorithm that excessive packet accumulation is happening, forcing to resend packets from the transport layer.

In our experiments, we define two scenarios (c.f. Fig. 4). In the first scenario, two IP flows belonging to two

3. The code used for these experiments can be found at https://github.com/mirazabal/Dynamic-buffer-TMC

	5G-BDP	DRQL	DynRLC[25]	CoDel[19]	USP
X-layer comm.	Yes	Yes	Yes	No	Yes
Pacing algoritm	Yes	No	No	No	Yes
Knob-less	Yes	Yes	No	No	Yes
Discard packets	No	No	No	Yes (or ECN)	No

TABLE 1
Different algorithms used to reduce the flows' latency.

QoS classes (i.e., two different QFIs) are mapped into 2 different SDAP queues that are lastly mapped into one RLC buffer. This is the case for services with different QFIs but sharing a RLC buffer. As mentioned in Section 2, services with different QFIs will inevitably share RLC buffers, and it is of capital importance to avoid the bufferbloat problem if different constraints of diverse services want to be fulfilled. In the second scenario, we mapped two different flows into one SDAP queue that is mapped into one RLC queue. Due to the very high number of flows that are nowadays generated and its dynamic nature, segregating all flows to distinct QFIs is unrealistic, due to the limited QFI range. Therefore, some non-critical low-latency traffic will end up in a scenario, where the created delays of shared queues may downgrade the user experience due to the bufferbloat. If the communication between adjacent sublayers/entities in the data path (e.g., UPF-SDAP) can be performed within a TTI, a pacing algorithm that gradually forwards the packets between these sublayers/entities can be implemented. This increases the chances of delivering a newly arrived delay constrained packet through the different queues without suffering big sojourn times. In contrast, a bulky algorithm forwards all the packets at once (i.e., an algorithm that lacks any pacing mechanism), and therefore, a newly arrived low-latency packet will suffer a considerable sojourn time, caused by the filled buffer with the previously forwarded packets from the lesser priority flows. Therefore, we implement the USP algorithm at the SDAP queues and combine it with the three methods aforementioned: DynRLC, DRQL and 5G-BDP at the RLC queues. We also study the BBR case, along with the vanilla case that represents the basic scenario described by 3GPP. Furthermore, in case where such a communication is not possible (e.g., splitted 5G scenarios or any standard setup where UPF and 5G-AN are separated), we place the well known AQM algorithm CoDel at the SDAP buffers and combine it with DynRLC, DRQL and 5G-BDP at the RLC buffer, in order to indicate to the transport layer's congestion control algorithm of the sender that excessive buffer accumulation is happening. A CoDel-CoDel combination is not presented as CoDel is not a backpressure mechanism and, thus, all the packets would flow to the RLC, leading to the same results as in the first scenario. The competing traffic flows (i.e., one with low-latency requirements and one with bursty traffic) emulate a scenario where different flows try to access the scarce resources in a saturated scenario. The low-latency traffic is modeled by UDP datagrams that

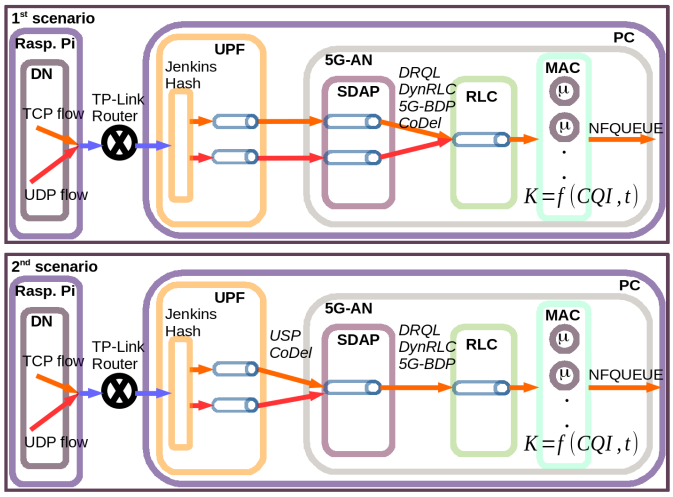

Fig. 4. Evaluation framework with 1st and 2nd scenarios.

emulate the traffic of online gaming applications [38].

Based on the traffic characteristics of such applications derived from real traces [39], we generated packets with a range of average interval times following a normal distribution and a standard deviation of 5 ms. The average interval times evaluated are [20,40,60,70] ms. The competing greedy flow is modeled using a TCP flow generated by the *iperf3* software with a MTU of 1500 bytes. The choice of TCP as the competing flow resides in the fact that most Internet traffic is forwarded through HTTP/2, which relies on TCP [40] as its transport layer protocol. In order to achieve the steady state and avoid the TCP slow start, the bursty traffic is generated 5 seconds before the UDP datagrams.

A PC and a Raspberry Pi are used to emulate the 5G QoS operations and the Data Network, respectively. The 5G QoS emulation PC contains an Intel(R) Core(TM) i7-7500U CPU @2.70GHz running Ubuntu 16.04, while the DN is implemented on a Raspberry Pi Model 3 B+ with a Broadcom BCM2837B0, Cortex-A53 64-bit SoC @1.4 GHz. The DN generates both traffic flows, while the QoS multiqueueing emulation software runs on the 5G QoS emulation PC. A TP-LINK TL-WR841N router connects the DN with the 5G QoS emulating PC through an Ethernet connection. In order to redirect the packets from the kernel space to the user space for QoS operations, we use the NFQUEUE traffic control *netfilter* queue binding. All the input packets are forwarded to the user space through an *iptables* rule, where the QoS solutions presented are applied, including the packet forward/drop decisions (e.g., AQM algorithms may decide to drop a packet in order to inform the congestion control TCP algorithm that excessive packet accumulation is taking place). Every sublayer/entity (i.e., UPF, SDAP and MAC) runs in a separate thread that executes in an infinite loop. All the code has been developed in C with efficiency and portability in mind. For realistic evaluations, we use LTE traces provided by [39], where the reported CQIs are converted to data link rates (i.e., they represent the K value from Fig. 4), that is a function of time and CQI as explained in Section 4. At [39], statistics from the base stations are reported in a granularity of 1 second for 15 minutes and five different cases (i.e., bus, car, train, pedestrian and static). From these 5 cases, we select the pedestrian and the train cases as they represent two completely different circumstances to which the network will be exposed. Since the conversion from the CQI index to the MCS index is manufacturer specific, we employed the well tested values from the OpenAirInterface project[4]. For the evaluations, the first 200 seconds of the traces are used, which correspond to 200 CQI updates and 20000 MAC scheduling opportunities per run (MAC scheduler egresses packets every 10 ms in our emulator). A large continuous simulation was chosen rather than many short ones in order to take into consideration large CQI variations, as well as to observe long term effects for some algorithms (e.g., BBR recalculates each state every 10 seconds).

When the flows arrive from DN to the 5G network emulation PC, they are hashed using the 5-tuple (source/destination IP address, source/destination port and type of traffic) as the key for the Jenkins hash function and classified as an IP tuple flow. We employ a SFQ [24], which minimizes a possible collision between different flows while providing an efficient solution.

The UPF network entity gets a transmission opportunity every 1 ms, where a maximum of 10 packets can be forwarded to the SDAP. As mentioned in Section 2, the SDAP sublayer maps UPF marked packets into RLC buffers based on their QFI. Since the egressing scheduler is not specified by 3GPP, we implement a Priority scheduler where low-latency packets are firstly dequeued, with a capability of egressing 10 packets among active SDAP queues and its priority every millisecond. The packets are finally forwarded into the RLC buffer, where they wait until the MAC sublayer pulls every 10 ms the amount of bytes requested by the PHY layer. We implement the MAC scheduler as a Round Robin scheduler, where the number of bytes that can be egressed is determined by the radio channel conditions according to [39]. Since the bottleneck resides at the MAC rather than at SDAP or UPF, it was decided to create a ten to one relationship between the forwarding capabilities of the UPF and SDAP in comparison with the MAC sublayer. The fact that the emulation was run on a soft real-time system, prevent the adoption of forwarding packets every 1 ms at MAC sublayer, which is the TTI for 4G cellular systems, and 0.1 ms at SDAP and UPF. Anyway, the results generated are expected to follow the same trend if the MAC sublayer forwards packets every 1 ms or 10 ms. With these parameter values, we successfully emulate the realistic scenarios where the packets tend to accumulate at the lower entities since the radio data link is the principal bottleneck of cellular systems. In our

4. Conversion function cqi_to_mcs can be found at openair1/PHY/phy_vars.h [26]

emulation scenario, and since the MAC egresses packets every 10 ms in discrete time, the *target time* of 5 ms recommended for CoDel [19] would never be reached when CoDel is implemented directly at the RLC buffer. Therefore, we increased the *target time* to 15 ms and the *interval time* to 300 ms, fulfilling the recommendation of maintaining the *target time* to 5% of the *interval time*. On the other hand, we have maintained its default values when the CoDel algorithm is used at the SDAP queues. The CoDel implementation has been implemented from scratch following the pseudo-code provided by [19].

For simulating the DynRLC [25] algorithm, we set the interval value to 100 ms and the desired sojourn time to 11 ms. A lower desired value would imply that the RLC buffer resides in a perpetual congestion state as the MAC scheduler egresses the packets every 10 ms. DynRLC adjusts its queue size through steps every interval time. The step size is determined by a α value that we set to 0.1. The MAX_VAL_LIMIT was set to 1024 packets for the DRQL and MAX_OCC was set to 1024 packets for the USP algorithm in order to discard an overflowing buffer effect from the study. The $NORMALIZED\_OFFSET$ was set to 0.33 in the 5G-BDP to avoid a starving fatal condition at RLC in a soft real-time environment as our emulation scenario. These values have been heuristically proved to be efficient.

6 Evaluation results

This section is divided into two parts. In the first part, we present the results of the first scenario, where two SDAP buffers are mapped into one RLC buffer. We name this first scenario *Individual SDAP buffer, shared RLC buffer*. In the second part, we present the results of the second scenario, and we name it *Shared SDAP buffer, shared RLC buffer*, where both flows share the SDAP as well as the RLC queue.

6.1 Individual SDAP buffer, shared RLC buffer

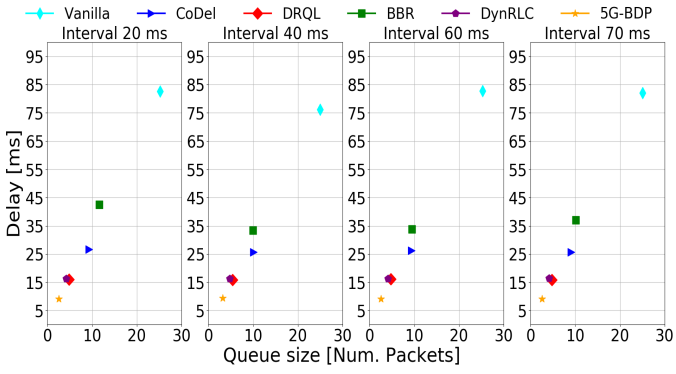

Fig. 5. Average queue size and average delay.

To quantitatively compare different QoS solutions, we assess the average delay of a low-latency packet from the moment that enters the UPF entity until it is forwarded by the MAC sublayer, and the average queue size of RLC when the low-latency packet is about to enter RLC. The results of the first scenario on the Pedestrian dataset are presented in Fig. 5 for the next algorithms: Vanilla (i.e., the basic scenario described by 3GPP with a priority scheduler at the SDAP sublayer), BBR, DynRLC, DRQL and 5G-BDP. Fig. 5 probes the correlation between the queue size and the delay a low-latency packet suffers. The algorithms that accumulate fewer packets at the RLC queue, suffer lower delays, while the algorithms that tend to accumulate more packets, suffer larger delays. This is a natural outcome from the discussion in Section 4.

In Fig. 6, the radio link channel capacity profile from the pedestrian dataset can be observed in red color (i.e., the previously represented K number of servers in Fig. 4). This profile represents the maximum number of packets that the PHY layer can pull. In green color, the size of the RLC buffer during the emulated 200 seconds can be observed with all the algorithms tested. If no countermeasures are applied (i.e., Vanilla case), excessive packet accumulation happens due to TCP's greedy congestion control nature, but no resources are squandered. The RLC buffer always possesses enough packets to forward to the MAC sublayer. A better result can be observed if CoDel is applied. In this case, the bandwidth is also nearly fully utilized, and the amount of packets accumulated at the RLC buffer is reduced. On the other hand, DRQL, DynRLC and 5G-BDP present a very similar RLC buffer occupancy graph. The three of them try to dynamically adjust RLC buffer size to the radio link channel capacity (i.e., graph in red). This ability enables them to come closer to the optimal queue size value where just enough packets to serve the PHY layer wait at the RLC buffer, but not more. Conversely, measuring the channel bandwidth at the senders congestion control algorithm is not as efficient as directly controlling the RLC queue size, as the BBR graph demonstrates.

In Section 4 at Fig. 3, we showed that there exist two inaccessible regions (i.e., minimum system response time and maximum utilization). The normalized utilization cannot surpass 1.0 and the minimum delay cannot go below 5 ms in average in our emulation scenario. The MAC scheduler pulls packets every 10 ms and, therefore, the optimal theoretical average time for an empty queue is 5 ms. These results come from the fact that the newly arrived packet may have to wait between [0-10] ms before it is forwarded to the next sublayer. Therefore, in the best case, would wait 0 ms, while in the worst case it would have to suffer a sojourn time of 10 ms in an empty queue. These theoretical results from Section 4 can be clearly observed in practice in Fig. 7 and Fig. 8. Both figures represent the empirical result of what we theoretically studied earlier in Section 4, where we deterministically came to the conclusion that no more data than the data accepted by the MAC should be kept at the RLC in any moment in order to work at the optimal β knee from the Fig. 3. From Fig. 7, 5G-

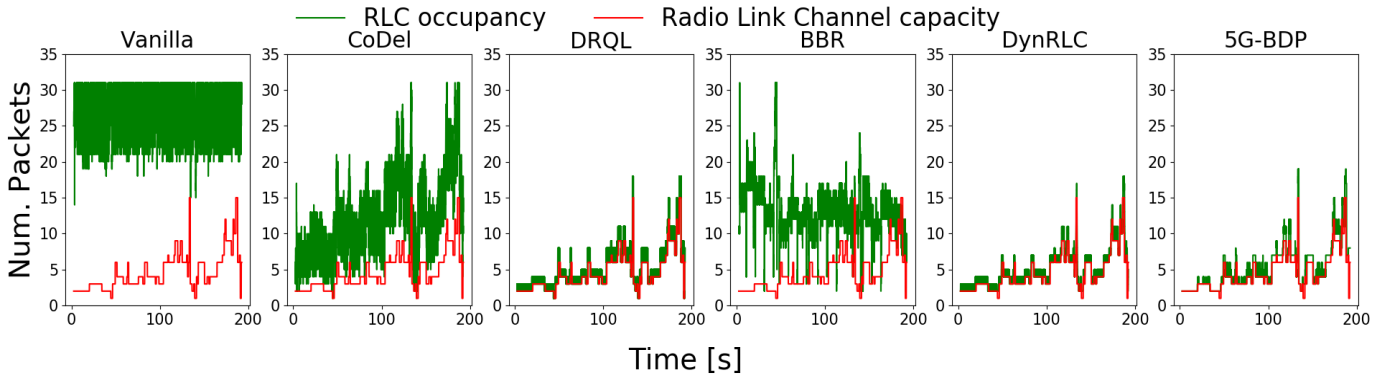

Fig. 6. Radio Link Channel Capacity and RLC queue occupancy.

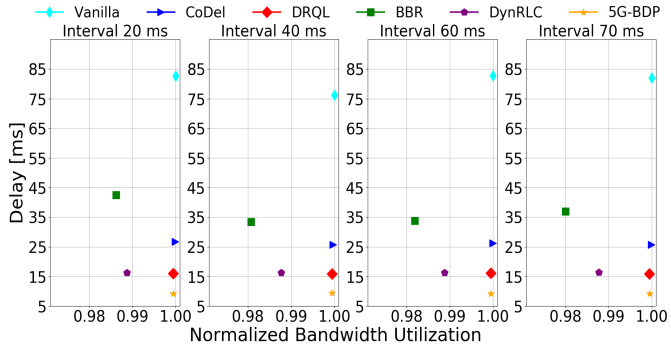

Fig. 7. First scenario, pedestrian dataset: Utilization rate and delay

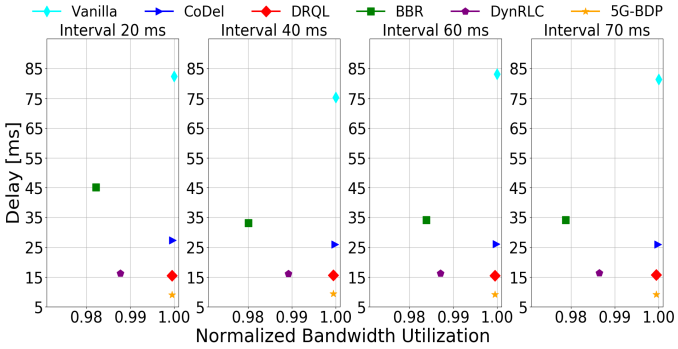

Fig. 8. First scenario, train dataset: Utilization rate and delay

BDP presents the best results as it maintains nearly full utilization (0.99947-0.99949) and presents the lowest delay (9.12-9.43) ms. The average delay is even below the 10 ms value used as the TTI for this emulation scenario. This value shows that packets have a nearly free path until the last queue (i.e., RLC buffers) and do not have to wait for a TTI on average. 5G-BDP achieves to work near the theoretical low limit (i.e., 5 ms in this scenario) while maintaining near full utilization thanks to the RLC queue size dynamicity and the pacing algorithm. On the other hand, BBR does not work within the granularity requested by 5G's cellular network stack and, therefore, fails as a candidate for delay-sensitive communications. BBR, recalculates the available bandwidth every 10 seconds as it can be seen from the BBR peaks at Fig. 6, and

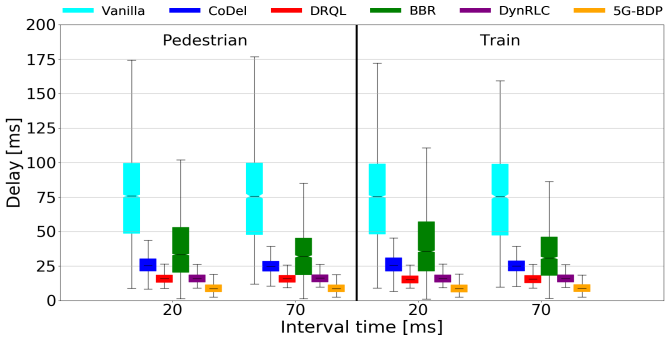

Fig. 9. Low-latency traffic boxplot for pedestrian and train datasets for interval times of 20 and 70 ms

adjusts the calculated bandwidth for the next 10 seconds. Such an approach lacks the ability to correctly adapt to the dynamic nature of 5G and, therefore, presents worse metrics than the other algorithms tested. BBR provides an utilization in the range of (0.98007-0.98619) and a delay in the range of (33.40-42.50) ms as it can be observed from Fig. 7. CoDel algorithm also exploits the communication between the sender and the receiver through TCP's congestion control algorithm discarding packets if congestion is detected. It needs some time until the congestion status information from the buffer can be transmitted to the sender through TCP's fast recovery mechanism. As seen in Fig. 7, it presents a normalized utilization that is in the range of (0.99967-0.99999) at a cost of a delay in the range of (25.66-26.69) ms. As demonstrated in the literature [29], CoDel works best if a little buffer is maintained after it, instead of collocating it directly at the edge. Since CoDel does not differentiate the packet type, 2.2% of the total low-latency packets were also discarded. Lastly, DRQL shrinks RLC's queue size in one step obtaining a normalized utilization range of (0.99994-0.99999) (c.f. Fig. 7) thus, slightly surpassing DynRLC that adapts its capacity through steps with a normalized utilization of (0.98774-0.98876). DRQL and DynRLC act directly on the queue without having to wait for additional information on transport delays, restraining the growth of the queue and ultimately reducing the delay. The delay ranges for DRQL and

DynRLC in Fig.7 are (15.92-16.11) ms and (16.23-16.40) ms, respectively. Thus using the pedestrian dataset, 5G-BDP reduces the delay in average by approximately 7.9 times with respect to Vanilla, 3.5 times compared to BBR, 2.7 times compared to CoDel and 1.7 times compared to DRQL and DynRLC. For the train dataset, the assessments from the pedestrian case hold as it can be observed in Fig. 8. Even though both datasets provide a different radio channel link profiles, both represent real models, and, as shown in Fig. 8, the algorithms proposed also adapt to different circumstances efficiently. In Fig. 9, the resulting delay values for the low-latency packets are presented with the lower and upper quartile values (i.e., 25th and 75th values). The horizontal line within the boxplots represents the median value and the whiskers represent the range of the data. From this figure, it can be extracted that the solutions that maintain the delay low, also present low jitter, which makes them appropriate for environments where not only the latency is considered, but also the variance of the latency is important.

In this subsection, Vanilla, CoDel, 5G-BDP, DRQL and DynRLC have been tested. It can be clearly seen that the Vanilla solution accumulates excessive number of packets in the bottleneck link in both scenarios, which leads to large average sojourn delays in the [75,85] ms range according to Fig. 7 and Fig. 8. CoDel, partially alleviates the bufferbloat through its ability to communicate with the sender's congestion control algorithm discarding packets. However, CoDel's approach still results in considerable average sojourn times (i.e., approx. 25 ms as shown in Fig. 7 and Fig. 8), and squanders some transmission opportunities. The approaches that segregate the traffic offer better results, as the low-latency traffic packets avoid the sojourn time at queues generated by greedy flows. Additionally, 5G-BDP's pacing capability allows it to avoid some sojourn time that, on the other hand, DRQL and DynRLC suffer from (i.e., low-latency packets with 5G-BDP experience delays of around 9 ms in average, against the around 16 ms in average of DynRLC and DRQL). DynRLC and DRQL lack of any pacing mechanism, and they forward packets at the beginning of the TTI, while 5G-BDP uniformly distributes the forwarding packets' process from the SDAP to the RLC in a TTI, thus, reducing the sojourn time for packets that arrive in the middle of a TTI. Moreover, DRQL and 5G-BDP, achieve the lowest standard deviation as shown in the results, being more stable in terms of jitter. All the methods tested present good resource utilization percentages (i.e., in the [0.97,1.0] range in accordance with Fig. 7 and Fig. 8). Both scenarios confirm the validity of the algorithms.

6.2 Shared SDAP buffer, shared RLC buffer

For the second scenario, as described in Section 5 and shown in Fig. 4, the congestion avoidance is more complex as the state of two levels of queues must be considered, and queues at upper entities have side effects on queues at lower entities. The RLC buffer must have enough bytes to forward once the MAC scheduler requests for transmission, but keeping additional bytes just adds delay. Moreover, the SDAP queues should have just enough bytes to feed the RLC buffer, but not more, while keeping the other packets at higher layers. The principle exposed by Kleinrock [33], which describes how data should be delivered at bottleneck's bandwidth speed in order to avoid starving the buffer and preventing forming queues, equally applies for the SDAP queues. From Fig. 10 and Fig. 11, it can be observed that the availability of a synchronous communication between the UPF and the SDAP is critical. The three methods tied with CoDel at SDAP queues present significantly worse results in delay, for both, train and pedestrian traces. It also shows that USP successfully feeds the SDAP queue with enough data to avoid a possible starvation at the RLC queue. From the moment where CoDel discards a packet due to excessive sojourn time until TCP's congestion control algorithm realizes that a packet was lost, the sender will send packets in a bursty manner. Even though such a solution may appear slow, it surpasses the BBR algorithm, where once again it clearly shows the limitations of tackling the bufferbloat problem in low-latency systems relying on the congestion control algorithm.

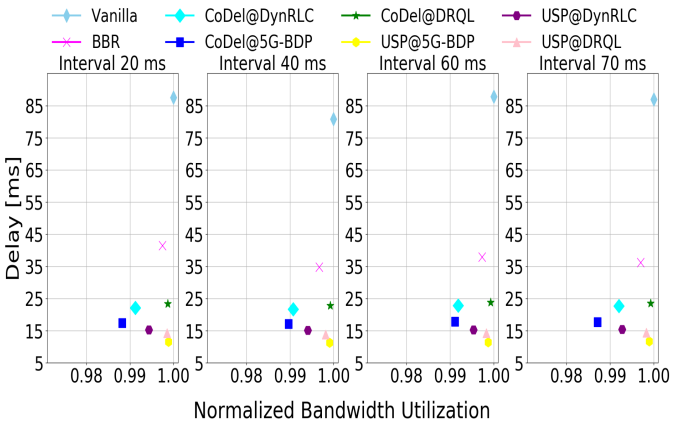

Fig. 10. Second scenario, pedestrian dataset: Utilization rate and delay

Again, the combination of USP with 5G-BDP outperforms all other alternatives due to its pacing nature in comparison with the bursty nature of USP with DynRLC or USP with DRQL. In particular, it approximately surpasses the other alternatives by 7.3, 3.1, 1.9, 1.5, 2.1, 1.4, 1.3 times in comparison with Vanilla, BBR, Codel@DynRLC, Codel@5G-BDP, CoDel@DRQL, USP@DynRLC and USP@DRQL, respectively. CoDel@DynRLC, Codel@5G-BDP and CoDel@DRQL dropped 2.4%, 3.8% and 2.5% of the total low-latency packets, respectively. A pacing algorithm such as 5G-BDP, maintains the packets during larger time in the queue in comparison with burst algorithms such as DynRLC or DRQL and, thus, the dropping percentage is slightly superior as CoDel interprets the

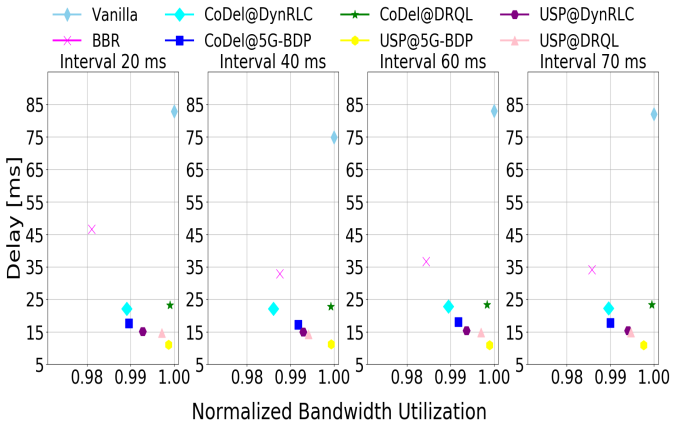

Fig. 11. Second scenario train dataset: Utilization rate and delay

sojourn time as a congestion symptom.

The results from this scenario show worse performance metrics than the results shown from the previous scenario. This is an expected result as, in this case, two queues need to be managed while in the earlier scenario, a single queue was managed. We also conducted experiments where two low-latency flows coexist with a background TCP flow for both scenarios. In these experiments both low latency flows were generated with a normal distribution and a 5 ms standard deviation. One was created with an average interval time of 50 ms, while the average interval for the second one was tested for [20,30,40,50,60,70] ms. The results obtained are very similar to the ones presented in this section quantitatively. Therefore, for the sake of brevity and due to the lack of novel outcome, these results are omitted. Such similarity in performance is expected for the evaluated solutions, since the amount of arrivals of two low-latency flows does not significantly impact the bulky flow's bandwidth utilization. Moreover, since the bufferbloat is created by TCP, the delay of the low-latency packets is predominantly governed by how the methods segregate the TCP and low-latency flows. Hence, no significant changes were observed regarding the delay with the addition of a second low-latency flow.

7 Conclusion

In this paper we have extensively emulated the actual 5G QoS system, through which we shed light into the bufferbloat problem with 5G specificities. We proved that the described vanilla case from 3GPP does not address the large sojourn times generated at 5G-AN buffers caused by excessive packet accumulation. We have proposed different solutions depending on the 5G scenario at hand. The most remarkable results are:

- Dynamic queue sizing and pacing: Based on queuing theory results [33] and 5G specificities, 5G-BDP has been designed and presented with very promising results. The utilization rate is maintained close to its maximum, while reducing the delay to nearly its minimum possible value. 5G-BDP successfully fulfils the challenges of maintaining the buffers at the optimum size, enabling a rapid packet delivery (at least a 70% latency reduction for the cases studied) and maintaining high utilization rate.
- SDAP queues: 3GPP should explicitly define a queuing system at the SDAP sublayer. Meticulously managing the queuing system at UPF will not successfully achieve envisioned requirements if the bufferbloat happens at lower stack entities. We also presented the USP algorithm that surpasses BBR and CoDel, and that can be combined with any RLC queue size limiting algorithm.

As the future work, wired (e.g., Time Sensitive Networks) and wireless low-latency hard real-time constraint systems should be studied jointly, as services rely on end-to-end delay rather than in partial solutions.

Acknowledgment

This work has been supported in part by the EU Horizon 2020 research and innovation program under grant agreement No. 675806 (5GAuRA), and in part by the Secretaria d'Universitats i Recerca del Departament d'Empresa i Coneixement from the Generalitat de Catalunya under grant agreement No. 2017 SGR 376.

References

[1] 3GPP, "System architecture for the 5G System," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 23.501, Dec. 2018, version 15.4.0.
[2] ——, "Evolved Universal Terrestrial Radio Access (E-UTRA) and NR; Service Data Adaptation Protocol (SDAP) specification," 3rd Generation Partnership Project (3GPP), Technical Report (TR) 37.324, Sep. 2018, version 15.1.0.
[3] The Bufferbloat Project. [Online]. Available: https://www.bufferbloat.net/projects/
[4] J. Gettys, "Bufferbloat: Dark Buffers in the Internet," *IEEE Internet Computing*, vol. 15, no. 3, pp. 96–96, May 2011.
[5] IEEE, "IEEE Standard for Local and Metropolitan Area Network–Bridges and bridged networks," *IEEE 802.1Q-2018*, July 2018.
[6] J. Corbet. (2012, Jul.) Tcp small queues. [Online]. Available: https://lwn.net/Articles/506237/
[7] T. Herbert. (2011, Nov.) Byte queue limits. [Online]. Available: https://lwn.net/Articles/469652/
[8] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "BBR: Congestion-based Congestion Control," *Commun. ACM*, vol. 60, no. 2, pp. 58–66, Jan. 2017.
[9] A. Azari, M. Ozger, and C. Cavdar, "Risk-aware resource allocation for urllc: Challenges and strategies with machine learning," *IEEE Communications Magazine*, vol. 57, no. 3, pp. 42–48, 2019.
[10] 3GPP, "Interface between the Control Plane and the User Plane nodes," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 29.244, Jun. 2019, version 16.0.0.
[11] J. Costa-Requena, A. Poutanen, S. Vural, G. Kamel, C. Clark, and S. K. Roy, "SDN-Based UPF for Mobile Backhaul Network Slicing," in *2018 European Conference on Networks and Communications (EuCNC)*, June 2018, pp. 48–53.
[12] S. Kekki, W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin, K. Wen, K. Kim, R. Arora, A. Odgers, L. Contreras, and S. Scarpina. MEC in 5G networks. [Online]. Available: https://www.etsi.org/images/files/ETSIWhitePapers/etsi_wp28_mec_in_5G_FINAL.pdf
[13] 3GPP, "NR, Radio Resource Control (RRC) protocol specification," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.331, Apr. 2019, version 15.5.1.
[14] ——, "NR, Radio Link Control (RLC) specification," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.322, Jan. 2019, version 15.4.0.

[15] The O-RAN Alliance. [Online]. Available: https://www.o-ran.org/
[16] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega, D. Aziz, and H. Bakker, "Network slicing to enable scalability and flexibility in 5g mobile networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, 2017.
[17] A. K. Paul, H. Kawakami, A. Tachibana, and T. Hasegawa, "An AQM based congestion control for eNB RLC in 4G/LTE network," in *2016 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, May 2016, pp. 1–5.
[18] S. Floyd and V. Jacobson, "Random Early Detection Gateways for Congestion Avoidance," *IEEE/ACM Transactions on Networking*, vol. 1, no. 4, pp. 397–413, Aug. 1993.
[19] K. Nichols, V. Jacobson, A. McGregor, and J. Iyengar, "Controlled Delay Active Queue Management," Internet Requests for Comments, RFC Editor, RFC 8289, Jan. 2018.
[20] Openwrt. [Online]. Available: https://openwrt.org/
[21] M. Shreedhar and G. Varghese, "Efficient Fair Queueing Using Deficit Round Robin," in *Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication*, ser. SIGCOMM '95. New York, NY, USA: ACM, 1995, pp. 231–242.
[22] "The Flow Queue CoDel Packet Scheduler and Active Queue Management Algorithm," RFC 8290, Jan. 2018.
[23] M. Ihlar, A. Nazari, and R. Skog. (2018) Low latency, high flexibility - Virtual AQM. [Online]. Available: https://www.ericsson.com/en/ericsson-technology-review/archive/2018/virtual-aqm-for-mobile-networks
[24] P. E. McKenney, "Stochastic Fairness Queueing," in *Proc. of IEEE Int. Conf. on Computer Communications*, June 1990, pp. 733–740 vol.2.
[25] R. Kumar, A. Francini, S. Panwar, and S. Sharma, "Dynamic control of RLC buffer size for latency minimization in mobile RAN," in *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, April 2018, pp. 1–6.
[26] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "OpenAirInterface: A Flexible Platform for 5G Research," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, p. 33–38, Oct. 2014.
[27] P. Imputato, N. Patriciello, S. Avallone, and J. Mangues-Bafalluy, "Smart Backlog Management to Fight Bufferbloat in 3GPP Protocol Stacks," in *2019 16th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Jan 2019, pp. 1–6.
[28] N. Bouacida and B. Shihada, "Practical and Dynamic Buffer Sizing Using LearnQueue," *IEEE Transactions on Mobile Computing*, vol. 18, no. 8, pp. 1885–1897, Aug. 2019.
[29] M. Irazabal, E. Lopez-Aguilera, and I. Demirkol, "Active Queue Management as Quality of Service Enabler for 5G Networks," in *2019 European Conference on Networks and Communication (EuCNC)*, April 2019, pp. 1–5.
[30] C. A. Grazia, N. Patriciello, T. Høiland-Jørgensen, M. Klapez, M. Casoni, and J. Mangues-Bafalluy, "Adapting TCP Small Queues for IEEE 802.11 Networks," in *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, Sep. 2018, pp. 1–6.
[31] H. Jiang, Y. Wang, K. Lee, and I. Rhee, "Tackling Bufferbloat in 3G/4G Networks," in *Proceedings of the 2012 Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 329–342.
[32] T. Herbert. (2011, Nov.) Dynamic queue limit. [Online]. Available: https://lwn.net/Articles/469651/
[33] L. Kleinrock, "Internet Congestion Control Using the Power Metric: Keep the Pipe Just Full, But No Fuller," *Ad Hoc Networks*, pp. 142–157, November 2018.
[34] ——, *Queuing Systems*. Wiley Interscience, 1975, vol. 1.
[35] ——, "Power and Deterministic Rules of Thumb for Probabilistic Problems in Computer Communications," in *Conference Record, International Conference on Communications*, Boston, Massachusetts, June 1979, pp. 43.1.1–43.1.10.
[36] 3GPP, "NR; Physical layer procedures for data," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.214, Sep. 2019, version 15.7.0.
[37] ——, "NR, Physical channels and modulation," 3rd Generation Partnership Project (3GPP), Technical Specification (TS) 38.211, Jun. 2019, version 15.6.0.
[38] X. Che and B. Ip, "Packet-level traffic analysis of online games from the genre characteristics perspective," *J. Network and Computer Applications*, vol. 35, pp. 240–252, 01 2012.
[39] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond Throughput: A 4G LTE Dataset with Channel and Context Metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference*, ser. MMSys '18. New York, NY, USA: ACM, 2018, pp. 460–465.
[40] M. Belshe, R. Peon, and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)," Internet Requests for Comments, RFC Editor, RFC 7540, 2015.

Mikel Irazabal received the M.Sc. degree in telecommunications engineering from the Universitat Politècnica de Catalunya, BarcelonaTech (UPC), in 2011. He is currently pursuing the Ph.D. degree with the Department of Network Engineering, UPC. He participated as an Early Stage Researcher (ESR) in the European funded Project Application-Aware User-Centric Programmable Architectures for 5G multi-tenant networks (5G-AuRA) and an Innovative Training Network (ITN) of the Marie Skłodowska-Curie Actions (MSCA). His main research interests include the areas of low-latency wireless communications and programmable wireless communication systems.

Elena Lopez-Aguilera received the M.Sc. and the Ph.D. degrees in telecommunications engineering from the Universitat Politècnica de Catalunya (UPC), in 2001 and 2008, respectively. She is currently an Associate Professor and a member with the Wireless Networks Group, Networks Engineering Dept., UPC. She has published papers in journals and conferences in the area of wireless communications and has been involved in projects with public and private funding. Her main research interests include the study of IoT enabling technologies and 5G networks. Her experience comprises QoS, radio resource management, location mechanisms, and wake-up radio systems.

Ilker Demirkol received the B.Sc., M.Sc., and Ph.D. degrees in Computer Engineering from Bogazici University, Istanbul, Turkey. He is currently an Assoc. Prof. in Dept. of Mining, Industrial and ICT Engineering with the Universitat Politecnica de Catalunya, where he works on wireless networks and wake-up radio systems. His research targets communication protocol development for the aforementioned networks, along with performance evaluation and optimization of such systems. He was a recipient of the 2010 Best Mentor Award from the Electrical and Computer Engineering Department, University of Rochester, NY.

Navid Nikaein is a Professor in Communication System Department at Eurecom. He received his Ph.D. degree in communication systems from the Swiss Federal Institute of Technology EPFL in 2003. Broadly, his research contributions are in the areas of experimental 4G-5G system research related to radio access, edge, and core networks with a blend of communication, computing, and data analysis with a particular focus on industry-driven use-cases. He is a board member of OpenAirInterface.org software alliance as well as the founder of the Mosaic-5G.io initiative whose goal is to provide software-based 4G/5G service delivery platforms.